DOD Internet Protocol

External Reference Specification

March 28, 1986

AUTHOR: _____
John R. Lyman III

TDRB APPROVAL: _____

AD&C APPROVAL: _____

PROPRIETARY NOTICE

The technical content set forth in this document is the property of Control Data and is not to be disseminated, distributed or otherwise conveyed to third parties without the express written permission of Control Data.

This document is to be used for planning purposes only. It does not include any explicit or implied commitments for any specific release dates or feature content. These matters are currently under active development, and as such are subject to revision and replanning as circumstances warrant.

RECORD OF REVISION

Revision	Description	Author	Date
01	Draft Version	JRL3	11/01/84
02	DoD Traceability and preTDRB	JRL3	02/01/85
03	Updated after IP walkthrough	JRL3	09/20/85
04	Minor data structure change	JRL3	12/13/85
05	Updated after coding	JRL3	02/25/86
06	ICMP support added	JRL3	03/28/86

Table of Contents

1.0 INTRODUCTION

1.0 INTRODUCTION

1.1 PURPOSE

This document describes the external specifications of the Internet Protocol (IP) module. IP is the Department of Defense (DoD) standard for what is called a level three protocol in the OSI Reference model. IP is currently being used by various networks, including ARPANET. IP has also been included in the BSD 4.2 version of UNIX.

In the CDCNET environment IP will be implemented as a level 3B module. IP will use the standard 3A software to interface with the real world. This will allow the IP module to make use of all of the standard CDCNET network solutions. The general flavor of the IP interface is similar to the Xerox internet interface.

1.2 REFERENCES

The following manuals contain material that either defines the operation of the IP module and the modules it interfaces to, or provides additional insight into the use of the IP module.

1)	RFC-791	SRI	Internet Protocol
2)	RFC-792	SRI	Internet Control Message Protocol
3)	MIL-STD-1777	DoD	Internet Protocol Standard
4)	RFC-793	SRI	Transmission Control Protocol
5)	MIL-STD-1778	DoD	Transmission Control Protocol
6)	ARH6866	CDC	Transmission Control Protocol ERS
7)	ARH6879	CDC	Intranet 3A ERS
8)	ARH7118	CDC	DoD IP static Routing ERS

2.0 SERVICE_OVERVIEW

2.1 SERVICES_PROVIDED

The services that the IP provides to the ULP can be divided into five basic groups; general, fragmentation/reassembly, SAP management, data transfer, and ICMP services. Each of these services is described in one of the following sections.

2.1.1 GENERAL SERVICES

The Internet Protocol (IP) is designed to interconnect packet-switched communication networks to form an catnet. The IP transmits blocks of data, called IP datagrams, from sources to destinations throughout the catnet. Sources and destinations are hosts located on either the same network or connected networks. Each IP datagram is an independent entity unrelated to any other IP datagram. The IP module does not create connections or logical circuits and has no mechanism to promote data reliability, flow control, or sequencing.

The IP module provides services to transport layer protocols and relies on the services of the lower layer network protocol (3A intranet). The IP module also provides a gateway function between networks.

An Upper Layer Protocol (ULP) passes data to the IP module for delivery. The IP module packages the data as an IP datagram and passes it to the local network protocol (3A intranet) for transmission across the local network. If the destination host is on a local network, IP sends the datagram through the network directly to that host. If the destination host is on a remote network, then IP sends the datagram to a local gateway. The gateway, in turn, sends the datagram through the next network to the destination host, or to another gateway. The sequence of IP modules handling the datagram in transit is called the gateway route. The gateway route is based on the destination internet address. The IP modules share common rules for interpreting internet addresses to perform internet routing.

2.0 SERVICE OVERVIEW
2.1.1 GENERAL SERVICES

The IP module adds an IP header block to the front of all data that it receives from the ULP to create an IP datagram. This header block contains information including: the source address, the destination address, and the network parameters. The format of this header will not be described in this document due to the excellent description in [3]. Sections 9.1 thru 9.3 in [3] describe each of the fields and its use.

Occasionally, a gateway IP module or destination IP will encounter an error during datagram processing. Errors detected are reported via the Internet Control Message Protocol (ICMP) which is implemented in the IP module.

The interface to the IP module provides the ULP with the ability to specify certain properties of the transmission service according to its needs. The network parameters fall into two categories: service quality parameters and service options. The service options include; security labeling, source routing, route recording, stream identification timestamping, and the don't fragment flag. The service options are handled exclusively by the IP module and are implemented in full. The service quality parameters include; precedence, transmission mode, reliability, and speed. The service quality parameters are not handled by the IP module. The IP Routing module may use some of these parameters when choosing a route. The 3A Intranet layer Does not accept any of these parameters.

2.1.2 FRAGMENTATION AND REASSEMBLY SERVICES

While traveling from source to destination, datagrams may need to traverse a network those maximum packet size is smaller than the size of the datagram. To handle this condition, the IP module provides the ability to fragment and reassemble datagrams. A gateway at the smaller-packet network fragments the original datagram into pieces, called datagram fragments, that are small enough for transmission. The IP module in the destination host reassembles the datagram fragments to reproduce the original datagram. This operation is totally transparent to the layers above the IP layer.

2.1.3 SAP MANAGEMENT SERVICES

All services provided by the IP module are accessed through a Service Access Point (SAP). A SAP is basically defined as a set of routines which determines the two way interface between the IP module and a user module. The IP SAP management routines are accessed through global entry points, their functions are described below.

Open_sap — The open SAP routine does a number of things. First, the protocol is checked; if the protocol specified is not being used then an entry is registered in the IP tables. Second, the addresses of the upper level protocol (ULP) indication routines are stored into the registered entry. Finally, the address of the IP send request routine and the SAPid are returned.

Close_sap — The close SAP routine releases a SAP currently in use and allows the associated protocol to be used in a subsequent open request.

2.1.4 DATA TRANSFER SERVICES

The data transfer service routines provide the ULP with the ability to send and receive datagrams from other homogeneous ULPs in the internet environment. The IP module does not necessarily deliver datagrams in the same order that they are sent, however, datagrams will be delivered to the destination ULP in the same form as sent by the source ULP. Datagrams may be discarded by the IP module when insufficient resources are available for processing, and datagrams lost or discarded by the network layer are not detected. The IP module is intended to insulate the ULP from the characteristics of the network, such as, the 3A address and the maximum 3A packet size.

Send_data — The send routine is provided by the IP module, and is called by the ULP to send a datagram out on the network.

Receive_data — The receive_data routine is provided by the ULP module, and is called by the IP to present a datagram received from the network.

2.1.5 INTERNET CONTROL MESSAGE SERVICES

The IP module also provides all of the services of the Internet Control Message Protocol (ICMP). These services are provided through an interface composed of three subroutines. The interface has been designed to provide complete service to all IP users. Two seperate indication routines are available to allow the user to specify the level of information that will be provided. The function of each routine is described below.

Send_icmp — This routine is provided by the IP module and allows the ULP to send the following ICMP datagrams. Any ULP can send these ICMP datagrams but, it is suggested that this routine be used with caution. When an echo, timestamp, or information request is sent the identifier field of the datagram is set to the sending protocol number to allow the response to be presented to the sender only.

- Network unreachable.
- Host unreachable.
- Protocol unreachable.
- Port unreachable.
- Fragmentation needed and DF bit set.
- Source route failed.
- Time-to-live exceeded in transit.
- Fragment reassembly time exceeded.
- Parameter problem.
- Source Quench.
- Network redirect.
- Host redirect.
- TOS/network redirect.
- TOS/host redirect.
- Echo request and reply.
- Timestamp request and reply.
- Information request and reply.

icmp_ind — This routine is provided by the ULP. The IP module will call this routine to present an indication of all ICMP datagrams that are received for this host. If the ULP does not require this information the address may be specified as NIL on the open_sap request. If

an echo, timestamp, or information reply is received it will be delivered to the protocol which is specified by the identifier field of the datagram. IF an echo, timestamp, or information request is received it will be delivered to all protocols. The following indications may be presented.

Network unreachable.
Host unreachable.
Protocol unreachable.
Port unreachable.
Fragmentation needed and DF bit set.
Source route failed.
Time-to-live exceeded in transit.
Fragment reassembly time exceeded.
Parameter problem.
Source Quench.
Network redirect.
Host redirect.
TOS/network redirect.
TOS/host redirect.
Echo request and reply.
Timestamp request and reply.
Information request and reply.

error_ind — Most ULPs will only need indications for ICMP datagrams that are directed to their specific protocol. This routine is provided by the ULP and is called by the IP module to present indications of ICMP datagrams direct to the ULP specific protocol. Not all ICMP datagrams will generate an indication, the following indications may be presented to the ULP.

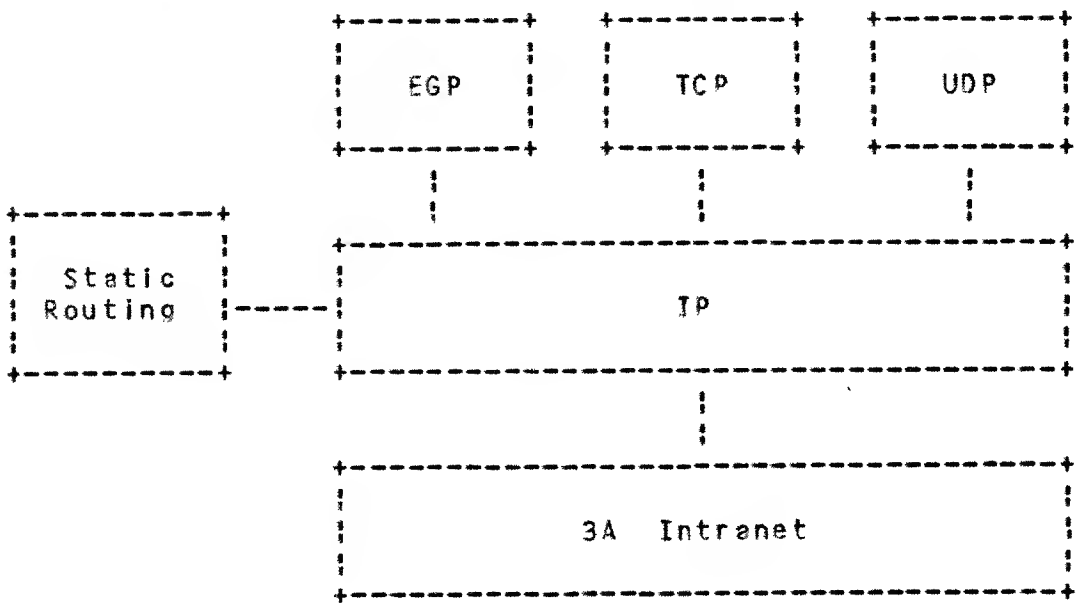
Network unreachable.
Host unreachable.
Protocol unreachable.
Port unreachable.
Fragmentation needed and DF bit set.
Source route failed.
Time-to-live exceeded in transit.
Fragment reassembly time exceeded.
Parameter problem.
Source Quench.

2.2 FUNCTIONAL RELATIONSHIPS

Before the IP data transfer services can be used an IP SAP must be opened. Data may then be transferred as long as needed. When all data transfers have been completed the SAP should be closed.

All IP SAPs will be opened by a user that is "well known" to all IP modules. This is because each SAP has a protocol identifier (PID) linked to it, and a PID can only be used by a single ULP. Unfortunately, there are only 254 PIDs available. For this reason, most users that need to exchange datagrams should work through the User Datagram Protocol (UDP) module instead of the IP module.

At this level, opening a SAP can be likened to picking up the phone to make a call. Once the phone has been picked up, a number of separate calls may be made and information may be transferred on each call. When finished with all calls the phone is hung up, this is similar to closing the SAP. Although this analogy is not quite perfect it does show the basic principles of the IP module. The following picture shows the modules that interface with the IP module.



2.3 EXTERNAL SERVICES UTILIZED

The following external interfaces are utilized by the IP module.

2.3.1 NETWORK LAYER

The network layer used by the IP module will be the 3A Intranet layer. The 3A module will provide transparent data transfer between the host within a single network. The only input required from the IP module should be the IP datagram, the network identification, and the local network address of the destination host. It is assumed that data will have a non-zero probability of arriving at the destination. If the network interface fails, then the 3A module is expected to notify the IP module.

The 3A module will provide a send routine which the IP module will call specifying the network identification, the local network address of the destination, and the data buffer. The data buffer will not be examined or changed by the 3A module.

The IP module will provide a receive routine which the 3A module will call to present data that it has received on an IP network. The 3A module will specify the network number, the local network address of the source, and the data buffer. The 3A module will not examine or modify the data buffer. The receive routine and the send routine are totally independent of each other and are asynchronous.

The IP will also provide a routine which the 3A module will call to inform the IP module that the status of a network connection has changed. This routine will receive the network number, the networks status, the maximum datagram size, and other unused information.

2.3.2 IP STATIC ROUTING MODULE

The IP Static Routing (IPSR) module is responsible for determining the next destination for all datagrams received by the IP module. The IPSR keeps track of the status of each network that IP can route datagrams to. The IPSR module will provide a routine which the IP module can call to obtain the destination of a datagram. The routine will accept the source address, the destination address, the network parameters, and the routing options from an IP datagram and return the next destination of the datagram and the maximum size of datagrams on that network.

3.0 SERVICE DESCRIPTIONS

3.0 SERVICE_DESCRIPTIONS

3.1 SAP_MANAGEMENT_SERVICES

3.1.1 DESCRIPTION

This section describes the routines that provide the SAP management services. A SAP is defined by a sapid and a set of routine addresses. The routines that provide SAP management are global addresses; all other services pass the addresses of their routines as parameters to and from the open_sap routine.

3.1.2 EXTERNAL INTERFACES

3.1.2.1 ip_open_sap_request

The open SAP routine is used to register a Service Access Point with the IP module. The IP module allows a total of 254 SAPs to be open at one time. Each SAP is identified by its SAPid value which is linked to the protocol value. The format of the CYBIL interface is as follows:

```
PROCEDURE ip_open_sap (
   protocol     : 0..255;
   data_ind     : ip_data_ind;
   error_ind    : ip_error_ind;
   icmp_ind     : ip_icmp_ind;
   VAR send_req : ip_send_req;
   VAR icmp_req : ip_icmp_req;
   VAR sapid    : INTEGER;
   VAR status   : ip_status_type);
```

protocol	In	This value specifies the user protocol to the IP module. The IP module will only allow one SAP for each protocol number. The ULP must specify an appropriate value between 1 and 254 inclusive.
data_ind	In	This is a pointer to a user supplied routine, which the IP module will call to present data messages to the ULP.

3.0 SERVICE DESCRIPTIONS
3.1.2.1 ip_open_sap request

error_ind	In	This is a pointer to a user supplied routine which the IP module will call to present ICMP indications that are specific to the protocol of the ULP. If the ULP doesn't need these indications then the pointer should be set to NIL.
icmp_ind	In	This is a pointer to a user supplied routine which the IP module will call to present all ICMP datagram indications. If the ULP doesn't need these indications then the pointer should be set to NIL.
send_req	Out	This is a pointer to the IP modules routine to send data.
icmp_req	Out	This is a pointer to a routine provided by the IP module to allow the ULP to generate ICMP datagrams. It is suggested that this request be used very carefully.
sapid	Out	This is a 32 bit value that identifies the particular IP SAP in all later requests. This parameter is used by the IP module as a password value.
status	Out	This is the status of the request. The following values may be returned: ip_insufficient_resources ip_protocol_illegal ip_protocol_inuse ip_successful

3.1.2.2 ip_close_sap_request

The close SAP routine is used to terminate the use of an IP SAP. This frees up the SAP for use by another ULP.

```
PROCEDURE ip_close_sap (
   protocol   : 0..255;
   sapid      : INTEGER;
   VAR status : ip_status_type);
```

3.0 SERVICE DESCRIPTIONS
3.1.2.2 ip_close_sap request

protocol	In	This value specifies the user protocol to the IP module.
sapid	In	This is the SAPid returned on the original open request.
status	Out	This is the status of the request. The following values may be returned: ip_protocol_illegal ip_sap_not_open ip_successful

3.2 DATA_TRANSFER_SERVICES

3.2.1 DESCRIPTION

The data transfer services are provided by two routines. The addresses of these routines are accessed through the open_sap routine. The send routine is provided by the IP module, and is called by the ULP module to send datagrams out onto the network. The indication routine is provided by the ULP module, and is used by the IP module to present datagrams received from the network. Note that a send_data request does not elicit any response, nor does a data_ind indicate that a send_data request is required. The parameters on the routines indicate how the data is to be packaged and where to send it, the data buffer is not examined or changed by the IP module.

3.2.2 EXTERNAL INTERFACES

3.2.2.1 ip_send_request

The send data routine is used to send data out on the network thru a previously opened SAP. The address of this routine is returned to the ULP when the open request is made.

```
ip_send_req = ^PROCEDURE (
   header      : ip_header_rec;
   source      : ip_address;
   destination : ip_address;
   options     : ^ip_option_rec;
   sapid       : INTEGER;
   VAR data    : buf_ptr;
   VAR status  : ip_status_type);
```

header	In	This is a record which contains the IP header for the datagram. The ULP module does not fill in the entire header only the user specified fields as noted in the data type section.
source	In	This is the address of the sender. This address may be partially or completely unspecified. This address is specified when a ULP

		needs to indicate which network solution the IP should use if it is multihomed.
destination	In	This is the address of the remote IP that the data is being sent to.
options	In	This is a pointer to a record which contains the IP options to be sent with the datagram.
sapid	In	This is the SAPid returned by the original open request.
data	In	This is a pointer to the system buffer containing the data portion of the datagram.
status	Out	This is the status of the request. The following values may be returned: ip_option_error ip_protocol_illegal ip_route_failed ip_sap_not_open ip_successful

3.2.2.2 ip_data_indication

The data indication routine is provided by the ULP, and is used by the IP module to present data messages to the ULP. The format of the Cybil interface is as follows:

```
ip_data_ind = ^PROCEDURE (
   header      : ip_header_rec;
   source      : ip_address;
   destination : ip_address;
   options     : ^ip_option_rec;
   sapid       : INTEGER;
   VAR data    : buf_ptr);
```

header	In	This is a record which contains the IP header from the datagram.
source	In	This is the address of the remote IP module that sent this datagram.

3.0 SERVICE DESCRIPTIONS
3.2.2.2 ip_data indication

destination	In	This is the address that the datagram was sent to. This address indicates which network solution the datagram was received on.
options	In	This is a pointer to a record which contains the IP options received with the datagram.
sapid	In	This is the SAPid returned by the original open request.
data	In	This is a pointer to the system buffer containing the data portion of the datagram.

3.3 INTERNET CONTROL MESSAGE SERVICES

3.3.1 DESCRIPTION

This section describes the routines which provide the interface to the ICMP functions supported by the IP module. All ULPs have the capability to send ICMP datagrams out on the network. This ability should be used with care. ULPs also have the option of receiving indications of all ICMP datagrams received from the network. The ULP selects this option by providing the address of an indication routine when it opens a SAP. Finally, the ULP can choose to receive indications about ICMP datagrams received from the network for its specific protocol. This option is also selected by providing the address of the ULPs indication routine when the SAP is opened. It is assumed that most ULPs will use this last indication service.

3.3.2 EXTERNAL INTERFACES

3.3.2.1 icmp_send_request

This routine is provided by the IP module to allow the ULP to send ICMP datagrams out to the network. This service must be used carefully to avoid flooding the network with useless ICMP datagrams. The format of the CYBIL interface is as follows.

```
ip_icmp_req = ^PROCEDURE (
   source      : ip_address;
   destination : ip_address;
   icmp_header : icmp_header_rec;
   protocol    : 0..255;
   sapid       : INTEGER;
   VAR data    : buf_ptr;
   VAR status  : ip_status_type);
```

source	In	This is the IP address of the sender within this DI.
destination	In	This is the IP address of the system which sent the datagram that is being rejected.

3.0 SERVICE DESCRIPTIONS
3.3.2.1 icmp_send request

icmp_header	In	This is the header portion of the ICMP datagram that the ULP is requesting the IP module to transmit.
protocol	In	This is the protocol number of the ULP requesting that an ICMP datagram be transmitted.
sapid	In	This is the SAPid allocated by the IP module when the SAP was opened.
data	In	This is a pointer to the system message buffer with contains any data that needs to go as part of the ICMP datagram.
status	Out	This is the status of the request. The following values may be returned. ip_destination_illegal ip_invalid_icmp_type ip_invalid_icmp_code ip_sap_not_open ip_source_illegal ip_successful

3.3.2.2 ip_icmp_indication

This routine is provided by the ULP and is called by the IP module when an ICMP datagram is received from the network. The ULP can refuse to accept these indications by specifing a NIL address on the open_sap request. The format of the Cybil interface is as follows:

```
ip_icmp_ind = ^PROCEDURE (
   source      : ip_address;
   destination : ip_address;
   icmp_header : icmp_header_rec;
   sapid       : INTEGER;
   VAR data    : buf_ptr);
```

source	In	This is the IP address of the host which sent this ICMP datagram out on the network.

destination	In	This is the IP address of an entity within this DI that the datagram pertains to.
icmp_header	In	This is the header portion of the ICMP datagram that was received from the network.
sapid	In	This is the SAPid allocated by the IP module when the SAP was opened.
data	In	This is a pointer to the system message buffer with contains any data that came with the ICMP datagram.

3.3.2.3 ip_error_indication

The error indication routine is provided by the ULP and is used by the IP module to present indications to the ULP which are protocol specific. The ULP can refuse to accept these indications by specifing a NIL address on the open_sap request. The format of the Cybil interface is as follows:

```
ip_error_ind = ^PROCEDURE (
   header      : ip_header_rec;
   source      : ip_address;
   destination : ip_address;
   options     : ^ip_option_rec;
   sapid       : INTEGER;
   data        : buf_ptr;
   error       : ip_status_type;
   bad_param   : INTEGER);
```

header	In	This is a record which contains the IP header from the datagram.
source	In	This is the address of the remote IP module that sent this datagram.
destination	In	This is the address that the datagram was sent to. This address indicates which network solution the datagram was received on.

options	In	This is a pointer to a record which contains the IP options received with the datagram.
sapid	In	This is the SAPid returned by the original open request.
data	In	This is a pointer to the system buffer containing the data portion of the datagram.
error	I	This is the type of error that occurred. The following errors are defined: ip_net_unreachable ip_host_unreachable ip_protocol_unreachable ip_port_unreachable ip_fragmentation_needed ip_route_failed ip_timeout ip_assembly_timeout ip_option_error ip_congestion
bad_param	IN	This is the identifier of the bad parameter in the case of an option error. The identifier may have the following values. 0 = Version or IHL 1 = Type of service 2 = Total length 3 = Identification 4 = Flags or fragment offset 5 = Time-to-live 6 = Protocol 7 = Checksum 8 = Source address 9 = Destination address 10 = Security option 11 = Streamid option 12 = Routing option 13 = Timing option

3.4 ERROR_RECOVERY

Although it can be assumed that the IP module will not be running in a hostile user environment, it is assumed that users will make mistakes. Therefore, the interfaces of the IP module are designed to catch errors as soon as possible. Most errors can be caught by simply checking parameters passed into the IP module. In terms of the IP module this is really error prevention rather than error recovery. The following list itemizes the steps that will be taken to catch errors.

1. Each SAP has a 32 bit SAPID to identify it. The sapid will be generated from a counter which will be updated each time a SAPID is used. This will prevent a user from trying to access a closed SAP and perhaps send data through a SAP reopened by another user.

2. All parameters will be tested for legality. The addresses passed in the open_sap request are the only exceptions to this rule. It is probably not possible to validate the addresses completely. The compiler will make sure that a parameter is the correct type, so other than checking for NIL pointers no checks will be made. Therefore, the IP module could possibly fail if those addresses are messed up.

3. There should be no error due to protocol that can make the IP module fail. If the 3A module passes bad information the error should be caught by parameter checks.

4.0 PERFORMANCE

4.1 OPERATING CHARACTERISTICS

A large number of other modules are dependent on the IP module. It is therefore necessary that the IP module be as efficient as possible. For every datagram that the IP module processes, a call is made to the IP routing module, therefore, part of the IP modules efficiency will depend upon the IP routing module. In general the following goals have been set.

Assume:
1) Transmission rate of 10meg bits/second.
2) Average packet size of 500 bytes.
3) Average 2 microseconds/68000 instruction.

Compute:
1) Average 2500 packets/second.
2) Average 400 microseconds/packet.
3) Approximately 200 instructions/packet.

4.2 OPERATIONAL MEASUREMENTS

The IP module accumulates a number of statistics which indicate the load that it has been working under. This information may be accessed through the standard CDNA Statistics Command Processor. The content and basic format of the information provided is shown below.

IP Statistics

Total Number of OPEN Requests = ########
Total Number of CLOSE Requests = ########
Number of SAPs Currently Open = ###

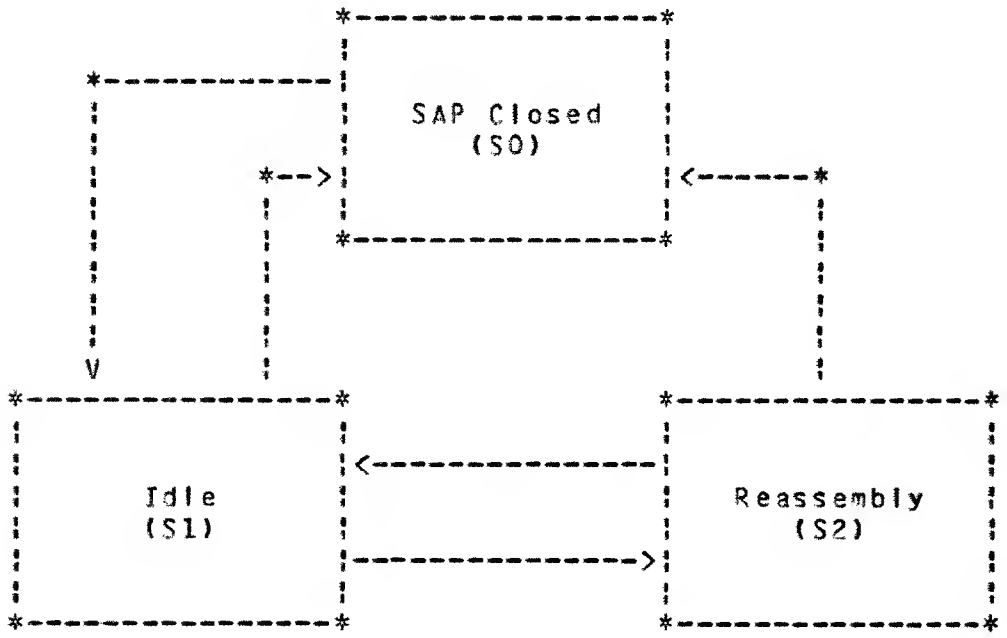
Protocol	Sent	Received	Packet errors
###	Packets= #######	Packets= #######	Resource= #######
	Bytes= #########	Bytes= #########	Content= #######
###	Packets= #######	Packets= #######	Resource= #######
	Bytes= #########	Bytes= #########	Content= #######
###	Packets= #######	Packets= #######	Resource= #######
	Bytes= #########	Bytes= #########	Content= #######
Total	Packets= #######	Packets= #######	Resource= #######
	Bytes= #########	Bytes= #########	Content= #######

5.0 FINITE STATE MACHINE

5.0 FINITE_STATE_MACHINE

The IP module will use a very simple FSM. The FSM is composed of three states, 12 input conditions, and 9 output actions. Each IP datagram will basically have it's own FSM. Each FSM instantiation will be identified by the source address, the destination address, the protocol, and the datagram identifier. The following sections describe the FSM.

5.1 FSM_DIAGRAM



5.2 FSM_INPUT_CONDITIONS

The following list contains the input conditions to the FSM. Each of the input conditions will, in combination with the current state, determine th next state and may produce an output action. Note that not all input conditions cause a change in the state, nor do all input conditions produce an output action.

(C1) A valid open request is received from the ULP.

(C2) A valid close request is received from the ULP.

(C3) A valid send request is received from the ULP.

(C4) A complete datagram is received from 3A.

(C5) A fragment of a datagram is received from 3A for a local destination.

(C6) A fragment of a datagram is received from 3A for a remote destination.

(C7) The assembly of a fragmented datagram is completed.

(C8) The timeout interval expires for a fragmented datagram under construction.

(C9) A datagram is received from 3A with a checksum error.

(C10) A datagram is received from 3A with an error other than a bad checksum.

(C11) A redirect error datagram is received from 3A.

(C12) A general error datagram is received from 3A.

5.3 FSM OUTPUT ACTIONS

The following list contains the output actions that the FSM produces. These output actions are produced when the input condition and the current state have the correct values.

(A1) Open the SAP.

(A2) Close the SAP.

(A3) Return the proper error status.

(A4) Determine the datagrams destination and send it to the ULP or 3A.

(A5) Discard the datagram.

(A6) Discard the datagram and send an error datagram to its source.

(A7) Begin or continue fragment collection.

(A8) Call the routing module to update the status of the route used.

(A9) Present the error datagram to the ULP.

5.4 FSM TRANSITION DEFINITIONS

This section shows what the next state and output action is for each pair of current state and input condition combinations. This table determines the total behavior of the FSM.

Input Condition	Input State S0	Input State S1	Input State S2
C1	S1,A1	S1,A3	S1,A3
C2	S0,A3	S0,A2	S0,A2
C3	S0,A3	S1,A4	S2,A4
C4	S0,A6	S1,A4	S2,A4
C5	S0,A6	S2,A7	S2,A7
C6	S0,A6	S1,A4	S2,A4
C7	S0,A5	S1,A5	S1,A4
C8	S0,A6	S1,A6	S1,A6
C9	S0,A5	S1,A5	S2,A5
C10	S0,A6	S1,A6	S2,A6
C11	S0,A8	S1,A8	S2,A8
C12	S0,A5	S1,A9	S2,A9

6.0 LOG MESSAGES

6.0 LOG_MESSAGES

The IP module issues log messages whenever it detects a function call error from either the ULP of the lower layer software. A log message will also be issued whenever a status command is processed, and whenever a configuration command is processed. The following is a list of each of the log messages generated.

6.1 PARAMETER_ERROR

Message Id: ???

Descriptive Message: "IP Parameter Error"

Fields: "Caller: " String(1..20)
"Function: " String(1..20)
"Parameter: " String(1..20)

Example: 84/10/31 11:29.35 123456789ABC
IP Parameter Error
Caller: Protocol_002
Function: Send_data
Parameter: Precedence

6.2 INTERNAL_ERROR_CONDITION

Message Id: ???

Descriptive Message: "IP Internal Error"

Fields: "Description: " String(1..60)

Example: 84/10/31 11:29.35 123456789ABC
IP Internal Error
Description: Data structure corrupted.

7.0 INSTALLATION OPTIONS

The IP module does not require any installation options. All parameters are either closely dependent on the code or must be set dynamically when the module is started or reconfigured.

8.0 NEW DATA TYPES

8.0 NEW_DATA_TYPES

Each user of the IP module is associated with an IP protocol type. There are a number of protocol types that are well know to IP modules through out the DoD network. The following protocol types are defined by the IP module and are stated to be well known to this IP module.

```
ip_protocol_ip  ......  Internal IP protocol.
ip_protocol_icmp  ....  Internal ICMP protocol.
ip_protocol_ggp  .....  Gateway to Gateway Protocol.
ip_protocol_tcp  .....  Transmission Control Protocol.
ip_protocol_udp  .....  User Datagram Protocol.
ip_protocol_cdcgw  ...  CDC gateway module.
```

All functions provided by the IP module return a status code. The following status codes are returned by one or more of the IP functions or the IP error indication routine.

```
TYPE
   ip_status_type = (
      ip_assembly_timeout,        { Timeout in reassembly.
      ip_congestion,              { No delivery, congestion.
      ip_destination_illegal,     { Illegal dest. address.
      ip_fragmentation_needed,    { Can't send without frag.
      ip_host_unreachable,        { Invalid host id.
    * ip_insufficient_resources,{ Memory/buffer congestion.  ;
      ip_invalid_icmp_type,       { ICMP type incorrect.       ;
      ip_invalid_icmp_code,       { ICMP code incorrect.       ;
      ip_net_unreachable,         { Invalid network id.
    * ip_option_error,            { Error in option.
      ip_port_unreachable,        { Invalid port.
    * ip_protocol_illegal,        { Illegal protocol number.
      ip_protocol_inuse,          { Requested protocol in use.
      ip_protocol_unreachable,    { Invalid protocol.
    * ip_route_failed,            { Errors in strict route.
    * ip_sap_not_open,            { The SAP is not open.
      ip_source_illegal,          { Illegal source address.
    * ip_successful,              { Operation completed.
      ip_timeout);                { Timeout during travel.
```

8.0 NEW DATA TYPES

The address record used by the IP interface is shown below. The "fields_inuse" field is used to indicate which of the following fields are specified. The only time that a field can be unspecified is when the address is the source address of a send_data request.

```
TYPE
   ip_address = RECORD
      fields_inuse : (ip_none, ip_net, ip_host, ip_both),
      network      : 0..0FFFFFF(16),
      host         : 0..0FFFFFF(16),
   RECEND,

   ip_double_word : -80000000(16)..7FFFFFFF(16);
```

The following record is used to describe the header used by the IP datagrams. This record is used by the ULP to pass the parameters that IP would have to put into the header any way. The only part of the header that IP processes in the source address and the destination address. The following CYBIL code defines the record and indicates which field the ULP must fill before calling IP.

```
TYPE
   ip_header_rec = RECORD
      version        : 0..15,              { ***
      ihl            : 0..15,              { ***
      precedence     : 0..7,               { ULP
      delay          : BOOLEAN,            { ULP
      throughput     : BOOLEAN,            { ULP
      reliability    : BOOLEAN,            { ULP                 !
      unused_flag_1  : BOOLEAN,            { **ZERO**            !
      unused_flag_2  : BOOLEAN,            { **ZERO**            !
      total_length   : 0..0FFFF(16),       { ***
      identification : 0..0FFFF(16),       { ULP                 !
      unused_flag_3  : BOOLEAN,            { **ZERO**            !
      dont_frag      : BOOLEAN,            { ULP
      more_frags     : BOOLEAN,            { ***
      frag_offset    : 0..01FFF(16),       { ***
      time_to_live   : 0..255,             { ULP
      protocol       : 0..255,             { ULP
      checksum       : 0..0FFFF(16),       { ***
      source         : ip_double_word,     { ***
      destination    : ip_double_word,     { ***
   RECEND;
```

8.0 NEW DATA TYPES

There are a number of options that may be included in the IP datagram. These options will specified by the user of the IP module. The following constants and types define the format of the options that are passed to and from the IP module. For more information about the use of options see the DoD specification MIL-STD-1777.

```
CONST
   ip_end           = 0,
   ip_no            = 1,
   ip_security      = 2,
   ip_loose_route   = 3,
   ip_timestamp     = 4,
   ip_record_route  = 7,
   ip_streamid      = 16,
   ip_strict_route  = 17,
   ip_max_option    = 31,

   ip_control_class = 0,
   ip_debug_class   = 2,
   ip_max_class     = 3,

   ip_loose  = 0,
   ip_record = 1,
   ip_strict = 2,

   ip_time_only    = 0,
   ip_record_addrs = 1,
   ip_stamp_addrs  = 2,


TYPE
   ip_option_rec = RECORD
      security : ip_security_rec,
      stream   : ip_streamid_rec,
      routing  : ip_routing_rec,
      timing   : ip_timing_rec,
   RECEND;

   ip_security_rec = RECORD
      inuse : BOOLEAN,
      s     : 0..0FFFF(16),
      c     : 0..0FFFF(16),
      h     : 0..0FFFF(16),
      tcc   : 0..0FFFFFF(16),
   RECEND,
```

8.0 NEW DATA TYPES

```
ip_streamid_rec = RECORD
   inuse : BOOLEAN,
   id    : 0..0FFFF(16),
RECEND,

ip_routing_rec = RECORD
   inuse : BOOLEAN,
   rtype : 0..255,     { ip_loose..ip_strict
   index : 0..255,     { 0..8
   count : 0..255,     { 0..9
   list  : ARRAY[0..8] OF ip_double_word,
RECEND,

ip_timestamp = RECORD
   inuse    : BOOLEAN,
   ttype    : 0..255,     { ip_time_only..ip_stamp_addrs
   index    : 0..255,     { 0..8
   count    : 0..255,     { 0..9
   overflow : 0..255,     { 0..15
   CASE 0..1 OF
      =0= sm_array : ARRAY[0..8] OF ip_double_word,
      =1= lg_array : ARRAY[0..3] OF ip_full_stamp,
   CASEND,
RECEND,

ip_full_stamp = RECORD
   addrs : ip_double_word,
   time  : ip_double_wrod,
RECEND;
```

The ICMP protocol uses IP datagrams to carry information. The header of the ICMP datagram is defined by the following record. The currently defined values for the TYPE and CODE fields are also defined below.

```
CONST
   icmp_echo_reply       = 0,
   icmp_dest_unreachable = 3,
   icmp_source_quench    = 4,
   icmp_redirect         = 5,
   icmp_echo_request     = 8,
   icmp_time_exceeded    = 11,
   icmp_parameter_error  = 12,
   icmp_time_request     = 13,
   icmp_time_reply       = 14,
   icmp_info_request     = 15,
   icmp_info_reply       = 16,
```

8.0 NEW DATA TYPES

```
      icmp_maximum_type      = 255,

      icmp_network_unreachable  = 0,
      icmp_host_unreachable     = 1,
      icmp_protocol_unreachable = 2,
      icmp_port_unreachable     = 3,
      icmp_fragmentation_needed = 4,
      icmp_source_route_failed  = 5,

      icmp_timeout           = 0,
      icmp_assembly_timeout = 1;


   TYPE
      icmp_header_rec = PACKED RECORD
         error_type : 0..255,
         error_code : 0..255,
         checksum   : 0..0FFFF(16),
         CASE 0..2 OF
            =0=
               gateway : ip_double_word,
            =1=
               ptr    : 0..255,
               unused : 0..0FFFFFF(16),
            =2=
               identifier : 0..0FFFF(16),
               sequence   : 0..0FFFF(16),
         CASEND,
      RECEND;
```

9.0 GLOSSARY

The following term definitions are provided to aid the reader in understanding this document.

CDNA: Control Data Distributed Network Architecture.

Datagram: This is a block of ULP data preceeded by an IP header and is the unit of data passed down by the IP module.

DoD: Department of Defense.

EGP: External Gateway Protocol. This protocol is used in the DoD network environment to pass simple routing information between individual catenets (see RFC-827, RFC-890, and RFC-904).

FIP: File Transfer Protocol. This is the protocol used in DoD systems to transfer files from host to host (see RFC-959).

IP: DoD Internet Protocol. This is the level three protocol used by the DoD. It provides the serves of an OSI internet layer.

OSI: Open System Interconnect. This is a model developed by the International Standards Organization (OSI) which defines a method of layering modules within networking software.

Protocol: Each user of the IP module is identified by a specific protocol identifier. This identifier is referred to throughout this document as the protocol.

SAP: Service Access Point. A SAP is defined in this manual as the point of access to the services that a module provides. In the case of the IP module this point of access is defined by a set of routine address provided by both sides of the service line.

9.0 GLOSSARY

ICP: DoD transmission control protocol. This is the transport protocol used by the DoD. TCP provides an error free orderly end to end data transmission (see MIL-STD-1778).

UDP: DoD user datagram protocol. This is a protocol which provides an single datagram transfer service with no promises (see RFC-768).

ULP: User Level Protocol. Used at various times to indicate the user of the module being discussed.